Delete Transform

> **NOTE:** Transforms are a part of the underlying language, which is not directly accessible to users. This content is maintained for reference purposes only. For more information on the user-accessible equivalent to transforms, see *Transformation Reference*.

Deletes a set of rows in your dataset, based on a condition specified in the `row` expression. If the conditional expression is `true`, then the row is deleted.

The `delete` transform is the opposite of the `keep` transform. See *Keep Transform*.

Basic Usage

```
delete row:(dateAge >= 90)
```

Output: For each row in the dataset, if the value in the `dateAge` column is greater than or equal to `90`, the row is deleted.

Syntax and Parameters

```
delete row:(expression)
```

Token	Required?	Data Type	Description
delete	Y	transform	Name of the transform
row	Y	string	Expression identifying the row or rows to delete. If expression evaluates to `true` for a row, the row is removed.

For more information on syntax standards, see *Language Documentation Syntax Notes*.

row

Expression to identify the row or rows on which to perform the transform. Expression must evaluate to `true` or `false`.

Examples:

Expression	Description
`Score >= 50`	`true` if the value in the `Score` column is greater than 50.
`LEN(LastName) > 8`	`true` if the length of the value in the `LastName` column is greater than 8.
`ISMISSING([Title])`	`true` if the row value in the `Title` column is missing.
`ISMISMATCHED(Score,['Integer'])`	`true` if the row value in the `Score` column is mismatched against the Integer data type.

Example:

```
delete row: (lastContactDate < 01/01/2010 || status == 'Inactive')
```

Output: Deletes any row in the dataset where the `lastContactDate` is before January 1, 2010 or the status is `Inactive`.

Usage Notes:

Required?	Data Type
Yes	Expression that evaluates to `true` or `false`

Examples

> **Tip:** For additional examples, see *Common Tasks*.

Example - Remove old products and keep new orders

This examples illustrates how you can keep and delete rows from your dataset using the following transforms:

- `delete` - Deletes a set of rows as evaluated by the conditional expression in the `row` parameter. See *Delete Transform*.
- `keep` - Retains a set of rows as evaluated by the conditional expression in the `row` parameter. All other rows are deleted from the dataset. See *Keep Transform*.

Source:

Your dataset includes the following order information. You want to edit your dataset so that:

- All orders for products that are no longer available are removed. These include the following product IDs: `P100`, `P101`, `P102`, `P103`.
- All orders that were placed within the last 90 days are retained.

OrderId	OrderDate	ProdId	ProductName	ProductColor	Qty	OrderValue
1001	6/14/2015	P100	Hat	Brown	1	90
1002	1/15/2016	P101	Hat	Black	2	180
1003	11/11/2015	P103	Sweater	Black	3	255
1004	8/6/2015	P105	Cardigan	Red	4	320
1005	7/29/2015	P103	Sweeter	Black	5	375
1006	12/1/2015	P102	Pants	White	6	420
1007	12/28/2015	P107	T-shirt	White	7	390
1008	1/15/2016	P105	Cardigan	Red	8	420
1009	1/31/2016	P108	Coat	Navy	9	495

Transformation:

First, you remove the orders for old products. Since the set of products is relatively small, you can start first by adding the following:

> **NOTE:** Just preview this transformation. Do not add it to your recipe yet.

Transformation Name	`Filter rows`
Parameter: Condition	`Custom formula`
Parameter: Type of formula	`Custom single`
Parameter: Condition	`(ProdId == 'P100')`
Parameter: Action	`Delete matching rows`

When this step is previewed, you should notice that the top row in the above table is highlighted for removal. Notice how the transformation relies on the `ProdId` value. If you look at the `ProductName` value, you might notice that there is a misspelling in one of the affected rows, so that column is not a good one for comparison purposes.

You can add the other product IDs to the transformation in the following expansion of the transformation, in which any row that has a matching `ProdId` value is removed:

Transformation Name	`Filter rows`
Parameter: Condition	`Custom formula`
Parameter: Type of formula	`Custom single`
Parameter: Condition	`(ProdId == 'P100' || ProdId == 'P101' || ProdId == 'P102' || ProdId == 'P103')`
Parameter: Action	`Delete matching rows`

When the above step is added to your recipe, you should see data that looks like the following:

OrderId	OrderDate	ProdId	ProductName	ProductColor	Qty	OrderValue
1004	8/6/2015	P105	Cardigan	Red	4	320
1007	12/28/2015	P107	T-shirt	White	7	390
1008	1/15/2016	P105	Cardigan	Red	8	420
1009	1/31/2016	P108	Coat	Navy	9	495

Now, you can filter out of the dataset orders that are older than 90 days. First, add a column with today's date:

Transformation Name	`New formula`
Parameter: Formula type	`Single row formula`
Parameter: Formula	`'2/25/16'`

Parameter: New column name	`'today'`

Keep the rows that are within 90 days of this date using the following:

Transformation Name	`Filter rows`
Parameter: Condition	`Custom formula`
Parameter: Type of formula	`Custom single`
Parameter: Condition	`datedif(OrderDate,today,day) <= 90`
Parameter: Action	`Keep matching rows`

Don't forget to delete the `today` column, which is no longer needed:

Transformation Name	`Delete columns`
Parameter: Columns	`today`
Parameter: Action	`Delete selected columns`

Results:

OrderId	OrderDate	ProdId	ProductName	ProductColor	Qty	OrderValue
1007	12/28/2015	P107	T-shirt	White	7	390
1008	1/15/2016	P105	Cardigan	Red	8	420
1009	1/31/2016	P108	Coat	Navy	9	495